APPLICATION FOR UNITED STATES LETTERS PATENT

FOR

TRACKING PROGRESS OF DATA STREAMER

Inventors: Zohar Bogin
Brent D. Chartrand
Arthur D. Hunter, Jr.

Prepared by: Jeffrey B. Huter
Patent Attorney

Attorney Docket No.: P17517

Intel Corporation
5000 W. Chandler Blvd., CH6-404
Chandler, AZ 85226-3699
Phone: (480) 554-4198
Facsimile: (480) 554-7738

"Express Mail" label number **EL 962028025 US**

TRACKING PROGRESS OF DATA STREAMER

BACKGROUND

[0001] PCI Express is an I/O (Input/Output) interconnect that attempts to be software compatible with PCI (Peripheral Component Interconnect). While trying to maintain some level of compatibility with PCI, PCI Express offers many features not found or fully supported by PCI. One such feature is the support of isochronous data transfers. Isochronous data transfer support may help multimedia applications such as, for example, audio and/or video playback applications achieve high quality results. However, the PCI Express implementation of isochronous data transfers has relaxed some of the stringent PCI ordering rules. As a result, a software application that isochronously transfers data via a PCI Express channel may need to utilize different techniques to track the progress of such a transfer than techniques used to track the progress of conventional PCI data transfers.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] The invention described herein is illustrated by way of example and not by way of limitation in the accompanying figures. For simplicity and clarity of illustration, elements illustrated in the figures are not necessarily drawn to scale. For example, the dimensions of some elements may be exaggerated relative to other elements for clarity. Further, where considered appropriate, reference labels have been repeated among the figures to indicate corresponding or analogous elements.

[0003] FIG. 1 illustrates an embodiment of a computing device with an audio controller.

[0004] FIG. 2 illustrates an embodiment of an audio controller in relation to buffers and buffer descriptor lists of a system memory. 1.

[0005] FIG. 3 illustrates an embodiment of a method of transferring data between a buffer in system memory and a codec.

DETAILED DESCRIPTION

[0006] The following description describes data streaming techniques. In the following description, numerous specific details such as logic implementations, opcodes, means to specify operands, resource partitioning/sharing/duplication implementations, types and interrelationships of system components, and logic partitioning/integration choices are set forth in order to provide a more thorough understanding of the present invention. It will be appreciated, however, by one skilled in the art that the invention may be practiced without such specific details. In other instances, control structures, gate level circuits and full software instruction sequences have not been shown in detail in order not to obscure the invention. Those of ordinary skill in the art, with the included descriptions, will be able to implement appropriate functionality without undue experimentation.

[0007] References in the specification to "one embodiment", "an embodiment", "an example embodiment", etc., indicate that the embodiment described may include a particular feature, structure, or characteristic, but every embodiment may not necessarily include the particular feature, structure, or characteristic. Moreover, such phrases are not necessarily referring to the same

embodiment. Further, when a particular feature, structure, or characteristic is described in connection with an embodiment, it is submitted that it is within the knowledge of one skilled in the art to effect such feature, structure, or characteristic in connection with other embodiments whether or not explicitly described.

[0008] Embodiments of the invention may be implemented in hardware, firmware, software, or any combination thereof. Embodiments of the invention may also be implemented as instructions stored on a machine-readable medium, which may be read and executed by one or more processors. A machine-readable medium may include any mechanism for storing or transmitting information in a form readable by a machine (e.g., a computing device). For example, a machine-readable medium may include read only memory (ROM); random access memory (RAM); magnetic disk storage media; optical storage media; flash memory devices; electrical, optical, acoustical or other forms of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.), and others.

[0009] An embodiment of a computing device is shown in FIG. 1. The computing device may comprise one or more processors 100 coupled and a chipset 102 via a processor bus 104. The chipset 102 may include one or more integrated circuit packages or chips that couple the processor 100 to a main or system memory 106, an audio controller 108, and/or other components 110 of the computing device. In particular, the chipset 102 may comprise one or more device interfaces 112 to support data transfers to and/or from other components

110 of the computing device such as, for example, BIOS firmware, keyboards, mice, storage devices, network interfaces, etc. via one or more buses 114.

[0010] The chipset 102 may further comprise a memory controller 116 to access system memory 106 via a memory bus 118. The memory controller 116 may access the system memory 106 in response to memory transactions associated with the processor 100, the audio controller 108, and other components 110 of the computing device. Further, the system memory 106 may comprise various memory devices that provide addressable storage locations which the memory controller 116 may read data from and/or write data to. In particular, the system memory 106 may comprise one or more different types of memory devices such as, for example, DRAM (Dynamic Random Access Memory) devices, SDRAM (Synchronous DRAM) devices, DDR (Double Data Rate) SDRAM devices, or other memory devices.

[0011] The computing device may further comprise one or more codecs 120 coupled to the audio controller 108 via an audio bus 122. The codecs 120 may be integrated into the audio controller 108 and/or chipset 102, may be mounted to a mainboard of the computing device, may be mounted to an add-in card that is coupled to the computing device, and/or may be part of an external device such as, for example, a docking station, audio mixer, etc that is coupled to an interface port (not shown) of the computing device. Further, the codecs 120 may be associated with sound cards, modems, facsimile devices, telephony devices, audio capture devices, video capture devices, etc. of the computing device that generate and/or process streams of data.

[0012] The audio controller 108 may stream data between the codecs 120 and buffers 124 of the system memory 106 defined by one or more buffer descriptor lists (BDL) 126 stored in the system memory 106. The audio controller 108 may also update a direct memory access (DMA) position in buffer (DPIB) structure 128 of the system memory 106 to reflect the progress of the audio controller 108 in transferring data between the audio controller 108 and the one or more buffers 124.

[0013] The audio controller 108 may be integrated into the chipset 102. However, in the depicted embodiment, the audio controller 108 is separate from the chipset 102. In such an embodiment, the chipset 102 and the audio controller 108 may each comprise one or more bus interfaces 130 that support isochronous data transfers across isochronous channels 132 and/or non-isochronous data transfers across non-isochronous channels 134. In one embodiment, one or more isochronous channels 132 couple a bus interface 130 of the audio controller 108 to a bus interface 130 of the chipset 102 to support isochronous data transfers therebetween. In one embodiment, each bus interface 130 may implement a PCI Express compatible interface that supports isochronous virtual channels. However, the bus interface 130 may implement additional and/or alternative interface protocols. Further, the chipset 102 and bus interface 130 may implement ordering rules similar to PCI Express ordering rules that are more lax than conventional PCI ordering rules. In particular, unlike PCI ordering rules, completion data for processor reads might not push isochronous reads and/or writes to system memory 106. Further, interrupts might not push

isochronous reads and/or writes to system memory 106. However, in one embodiment, the ordering rules ensure that isochronous writes do not pass previously issued isochronous reads and/or writes across the same channel. FIX-ME: ARE THESE ORDERING RULES ACCURATE?

[0014] The audio controller 108 may further comprise an audio bus interface 136 for an audio bus 122 used to couple the codecs 120 to the audio controller 108. In one embodiment, the audio bus interface 136 may receive frames of data from the codecs 120 via one or more point-to-point serial input links of the audio bus 122. The audio bus interface 136 may further send frames of data to one or more of the codecs 120 via a broadcast serial output link of the audio bus 122. that encode and/or decode streams in accordance to various formats.

[0015] Referring now to FIG. 2, the audio controller 108 may comprise one or more BDL DMA controllers 138, a DPIB DMA controller 140, one or more input DMA controllers 142, and one or more output DMA controllers 144. In one embodiment, the each input DMA controller 142 may be programmed by the processor 100 to separately stream data received from one or more codecs 120 to a buffer 124 of the system memory 106. Similarly, each output DMA controller 142 may be programmed by the processor 100 to separately stream data from a buffer 124 of the system memory 106 to one or more codecs 120.

[0016] Each BDL DMA controller 138 may read a buffer descriptor list 126 from the system memory 106 and provide the buffer descriptor list to the appropriate DMA controller 142, 144. Each buffer descriptor list 126 may define a plurality of buffer segments 146 of a buffer 124 from which a corresponding

output DMA controller 142, 144 may read data or from which a corresponding input DMA controller may write data. Each descriptor 148 of a buffer descriptor list 126 may comprise a base that identifies the start of a buffer segment 146 of the buffer 124 and a length that identifies the end of a buffer segment 146. In one embodiment, the DMA controllers 142, 144 may treat their respective buffers 124 as ring buffers or cyclic buffers. Accordingly, upon reaching the end of their respective buffers 124, the DMA controllers 142, 144 may wrap-around or return to the beginning of their corresponding buffer 124.

[0017] The audio controller 108 may further comprise a separate buffer length register 150, a separate DMA position in buffer (DPIB) counter 152, and a separate link position in buffer (LPIB) counter 154 for each input DMA controller 142 and output DMA controller 144. The processor 100 and/or the BDL DMA controllers 138 may store in the buffer length register 150 of each DMA controller 142, 144 the length of its corresponding buffer 124. Each input DMA controller 142 may increment its DPIB counter 152 such that the DPIB counter 152 contains a count equal to the number of bytes the input DMA controller 142 has transferred toward its buffer 124. Further, each input DMA controller 142 may increment its LPIB counter 154 such that its LPIB counter 154 contains a count equal to the number of bytes the input DMA controller 142 has received from its input link. Similarly, each output DMA controller 144 may increment its DPIB counter 152 such that the DPIB counter 152 contains a count equal to the number of bytes the output DMA controller 144 has transferred from its buffer 124. Further, each output DMA controller 144 may increment its LPIB counter

154 such that its LPIB counter 154 contains a count equal to the number of bytes the output DMA controller 144 has transferred toward its output link of the audio bus 122.

[0018] The DPIB counters 152 and LPIB counters 154 may clear themselves or may be otherwise reset to an initial value (e.g. zero, its corresponding buffer length, etc.) upon their count indicating that the end of the respective buffer 124 has been reached. In one embodiment, the DPIB counters 152 and LPIB counters 154 may clear themselves or may be otherwise reset to an initial value upon their count having a predetermined relationship to the buffer length of the corresponding buffer length register 150. In another embodiment, the DPIB counters 152 and LPIB counters 154 may clear themselves or may be otherwise reset to an initial value in response to the counters 152, 154 overflowing and/or underflowing. Further, the LPIB counters 154 may be implemented such that the processor 100 may read the count of the LPIB counters 154 but may not write to or update the count of LPIB counters 154.

[0019] The audio controller 108 may further comprise an update counter 156 and an update rate register 158. In one embodiment, the update counter 156 may generate an overflow signal in response to the count of the update counter 156 incrementing past a value stored in the update rate register 158. In response to the overflow, the count of the update counter 156 may be cleared or set to zero. In another embodiment, the update counter 156 may generate an underflow signal in response to the count of the update counter 156

decrementing past zero. In response to underflowing, the count of the update counter 156 may be set to the value stored in the update rate register 158.

[0020] The DPIB DMA controller 140 may use the underflow/overflow signals to determine when to update a DPIB structure 128. As depicted in FIG. 2, the DPIB structure 128 may comprise a DPIB field 160 for each input DMA controller 142 and each output DMA controller 144. The DPIB structure 128 may further comprise reserved fields 162 that are reserved for possible future use. In one embodiment, the DPIB structure 128 comprises a contiguous portion of system memory 106 and each field 160, 162 of the DPIB structure comprises 32 bits.

[0021] An embodiment of a method to stream or transfer data between a buffer 124 and a codec 120 is shown in FIG. 3. In box 200, the processor 100 may allocate a buffer 124 to a stream and may store a buffer descriptor list 126 in the system memory 106 that identifies the buffer segments 146 of the allocated buffer 124. The processor 100 in box 202 may allocate a DMA controller 142, 144 of the audio controller 108 to the stream and may configure the audio controller 108 to stream the data. In one embodiment, the processor 100 may provide the audio controller 108 with the length of the allocated buffer 124, the base address of the buffer descriptor list 126, a stream identifier of the stream to be transferred, a DMA identifier for the DMA controller 142, 144 to stream the data, one or more codec identifiers for the codecs 120 involved in the data transfer, and an update value for the update rate register 158.

[0022] In box 204, the audio controller 108 may configure one or more DMA controllers 138, 140, 142, 144 for the stream based upon the information

received from the processor 100. In one embodiment, the audio controller 108 may update the buffer length register 150 for the allocated DMA controller 142, 144 based upon the received buffer length, may initialize the DPIB counter and the LPIB counter 154 for the allocated DMA controller 142, 144, and may provide the BDL DMA controller 138 for the allocated DMA controller 142, 144 with the base address of the buffer descriptor list 126 for the allocated buffer 124 and the DMA identifier for the allocated DMA controller 142, 144.

[0023] The BDL DMA controller 138 in box 206 may read the buffer descriptor list 126 from the system memory 106. The BDL DMA controller 138 may further provide the buffer descriptor list 126 to the DMA controller 142, 144 associated with the received DMA identifier and/or may configure the DMA controller 142, 144 to transfer data per the read buffer descriptor list 126. The audio controller 108 in box 208 may then transfer data between the buffer 124 and a codec 120 per the buffer descriptor list 126. In particular for an input stream, the audio bus interface 136 may receive from a codec 120 the stream identifier and associated data via an input link of the audio bus 122. The audio bus interface 136 may route the data to the appropriate input DMA controller 142 based upon the stream identifier and the input DMA controller 142 may write the data to the allocated buffer 124 across an isochronous channel 132. Similarly, an output DMA controller 144 may read data across an isochronous channel 132 from the allocated buffer 124 as defined by the buffer descriptor list 126. The audio bus interface 136 may receive the data from the output DMA controller 144 and may

transfer the data in frames to the appropriate codecs 120 via an output link of the audio bus 122.

[0024] In response to sending or receiving data via the audio bus interface 136, the DMA controller 142, 144 in box 210 may update its respective LPIB counter 154 to reflect the progress of the transfer on the link of the audio bus 122. In one embodiment, the DMA controller 142, 144 may increment its LPIB counter 154 by the number of bytes transferred on the link since the last update. Further, the DMA controller 142, 144 may update in box 212 the update counter 156 based upon data transferred on its link. In one embodiment, each DMA controller 142, 144 may update the count of the update counter 156 by the number of frames transferred on its respective link since its last update thus causing the update counter 156 to track the number of frames transferred on the audio bus 122.

[0025] In response to reading from or writing data to the buffer 124, the DMA controller 142, 144 in box 214 may update its respective DPIB counter 152 to reflect the progress that the DMA controller 142, 144 has made in transferring data between its buffer 124 and the audio controller 108. In one embodiment, the DMA controller 142, 144 may update its DPIB counter 152 by the number of bytes transferred between its buffer 124 and the audio controller 108 since the last update.

[0026] In box 216, the DPIB DMA controller 140 may determine whether to update the DPIB structure 128 in the system memory 106 based upon the count of the update counter 156. In one embodiment, the DPIB DMA controller 140

may determine to update the DPIB structure 128 stored in the system memory 106 in response to an overflow and/or underflow signal of the update counter 156. In another embodiment, the DPIB DMA controller 140 may determine to update the DPIB structure 128 in response to determining that the count of the update counter 156 has a predetermined relationship (e.g. equal) to an update value stored in the update rate register 158. In response to determining to update the DPIB structure 128, the DPIB DMA controller 140 in box 218 may write the current values of the DPIB counters 152 across the same virtual channel used by the DMA controllers 142, 144 to their respective DPIB fields 160 of the DPIB structure 128 in the system memory 106.

[0027] As a result of the bus interfaces 130 maintaining write ordering across the virtual channel, the processor 100 may determine from the DPIB structure 128 the progress of the audio controller 108 in isochronously transferring data to and/or from a buffer 124 in the system memory 106. Further, the processor 100 may read the LPIB counter 154 to determine the progress of the audio controller 108 in transferring data across a link of the audio bus 122.

[0028] In box 220, the audio controller 108 may determine whether the DMA controller 142, 144 has reached the end of its buffer 124. In one embodiment, the audio controller 108 may determine that the DMA controller 142, 144 has reached the end of its buffer 124 in response to an overflow and/or underflow signal of the DPIB counter 156. In another embodiment, the audio controller 108 may determine that the DMA controller 142, 144 has reached the end of its buffer 124 in response to determining that the count of the DPIB counter 152 has a

predetermined relationship (e.g. equal) to a buffer length stored in its corresponding buffer length register 150. In response to determining that the end of the buffer 124 has been reached, the DMA controller 142, 144 in box 222 may reset or initialize its DPIB counter 152 and LPIB counter 154. The DMA controller 142, 144 may further return to the start of its buffer 124 per its buffer descriptor list 126. The DMA controller 142, 144 may then return to box 208 to continue transferring data between its buffer 124 and the audio controller 108.

[0029] Certain features of the invention have been described with reference to example embodiments. However, the description is not intended to be construed in a limiting sense. Various modifications of the example embodiments, as well as other embodiments of the invention, which are apparent to persons skilled in the art to which the invention pertains are deemed to lie within the spirit and scope of the invention.